面向集成电路可靠性挑战的多核处理器虚拟化技术

张磊 韩银和 李晓维

摘要：多核处理器以其高性能、低功耗、设计周期短等诸多优势成为未来高性能处理器的发展趋势。由于应用对计算能力的需求是无限的，随着芯片上晶体管数目的进一步增多，多核处理器将逐渐过渡到大规模多核处理器或者称为众核处理器。多核处理器面临着很多的设计挑战，其中可靠性问题尤其严重。一方面，由于多核处理器的芯片面积都比较大，生产缺陷导致的成品率损失问题严重。这使得芯片上可能存在失效的处理器核，而且不同芯片上失效核的位置和分布也不相同。另一方面，工艺扰动问题使得多核处理器上各个处理器核的性能也存在差异。芯片上处理器核的失效以及性能差异使得不同芯片的底层结构各不相同，这给上层的操作系统和软件优化带来了负担。我们借助虚拟化的思想，将缺陷和核间性能差异对软件层进行屏蔽，提供统一的接口和界面，便于编程开发和管理。

关键词：众核处理器，片上网络，缺陷容忍，性能碎片，虚拟化

1 前言

按照摩尔定律，芯片上可以容纳的晶体管数目每 18 个月便会增加一倍。处理器设计师们很好地利用了这些丰富的片上资源，通过体系结构的不断创新，如精确的分支预测技术、猜测和乱序执行技术等等来提升微处理器性能。指令流水线不断加深，并行度不断增加，芯片的工作频率也不断提高。然而，传统的提高处理器性能的方法也使得设计和制造的复杂度越来越高，芯片的功耗越来越大，变得不可接受。因此，芯片厂商放弃花费高昂代价继续提高单处理器性能，开始转向在芯片上集成多个处理器核，通过并行计算提升处理器性能[1]。

应用对计算能力的需求是无限的，如科学计算、天气预报、基因工程、网络、多媒体等等。计算能力的大小直接决定了应用可以达到的规模和精度。芯片上晶体管数目的不断增多将使得多核处理器（Multi-core processor）逐渐过渡到众核处理器（Many-core processor）[2]。英特尔在 2006 年的开发者论坛上展示了一款含有 80 个简单处理器核[3]工作在 3.1GHz 的芯片原型。根据预测，到 2012 年这种片上超级计算机的性能可以达到每秒万亿次操作。以前需要安放在一个房间的超级计算机，现在可以集成在一个芯片上了。

大规模多核处理器系统由三部分组成：逻辑、存储和通信。在半导体工艺如此先进的今天，通信已经成为决定数字系统性能的关键因素。芯片的大部分功耗都用来驱动互连线，大部分时钟周期都花费在线延迟而非门延迟上。工艺的进步使得逻辑部件和存储器变得更小、更快、更便宜，而与之相比，引脚和互连线的集成度发展却很缓慢。随着计算和存储部件的不断增多，传统的片上通信方法，如共享总线，由于可扩展性很差而出现严重的性能退化。而另一种系统级的片上通信解决方案－“片上网络”（Network-on-Chip，简称 NoC）开始得到广泛的关注[4, 5]。片上网络借鉴了计算机网络和并行多处理器互连网络的技术，将计算和通信分离，和总线相比具有很好的可扩展性，而和专用的互连结构相比，成本更低，效率更高。图 1 所示为基于片上网络的多核处理器体系结构。

多核处理器的出现给系统设计人员以及应用程序开发人员带来了巨大的挑战。为了能不断地提高处理器性能，使其与摩尔定律带来的丰富资源相称，软件将承担更多的责任。例如，操作系统、编译器和应用软件需要显式地管理片上处理器核，开发更多的并行性以使得更多的处理器核忙碌。其中，片上处理器核之间的“动态异构性”将成为未来多核芯片的重要设计挑战，同时也提供了更多可以利用的空间[6, 11]。所谓动态异构性是指片上处理器核具有不同的，而且可能不断变化的性能和特点（即使处理器核被设计成同构的）。动态异构性主要来自于可靠性方面的挑战，如永久性、间歇性或瞬时性故障，工艺扰动，片上功耗和温度管理等。这使得软件所看到的处理器核可能具有不同的性能（频率），甚至软件可用的处理器核的数目也是不断变化的。例如当处理器核发生永久性故障时，操作系统将无法为其分配任务，当出现间歇性故障时，处理器核将会在一段时间内处于离线（off-line）状态，无法使用。此外芯片上可能会出现局部功耗过高，或者温度超过阈值，片上的动态功耗和热管理模块就会关闭某些处理器核，以达到降低功耗或降低温度的目的。

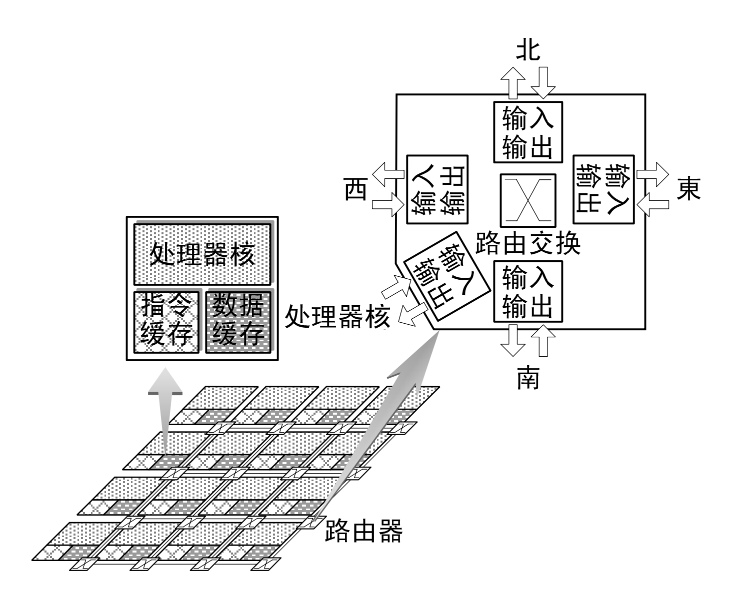


图1. 基于片上网络的多核处理器体系结构

另一方面，软件在使用片上处理器核的时候，需求也是灵活多变的。例如，金融安全方面的应用需要三模或者多模冗余来保证系统的可靠性和可用性，而对于网络以及多媒体应用来说则可以容忍更多的故障，从而换取功耗上的降低。多核处理器必须同时支持这些需求完全不同的应用，在逻辑三模冗余（即将三个核看作一个逻辑核）以及每个核单独使用之间进行切换。

片上处理器核之间的异构性使得计算和执行计算的物理硬件之间映射关系变得复杂。这种复杂性表现在在系统运行的任意时刻，计算或任务应该如何有效地映射到硬件资源上。处理这种动态的异构性并合理地加以利用需要详细地了解底层每一个处理器核的配置以及状态，而目前的大多数系统和应用软件是不掌握这些细节信息的。另一方面，硬件的配置和计算能力的变化是非常迅速的。如果操作系统跟踪记录底层硬件的变化并做出调度决定，与所要执行的任务相比，其开销非常大。因此在多核处理器中，利用操作系统和应用软件来处理硬件的变化和细节是不合适的，硬件本身应该更多地承担这种映射的任务，为操作系统和应用软件提供一个清晰的界面，减轻多核处理器系统软件和应用软件的设计负担。

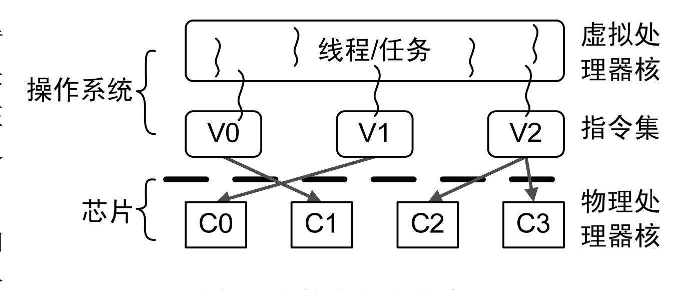


图2. 多核虚拟化技术

图 2 所示的多核虚拟化技术，通过硬件或固件将底层的硬件细节进行抽象，通过软硬件接口（如指令集）为操作系统和程序员提供数目固定的简单同构的处理器核。这种软件透明的虚拟化技术为上层用户提供了一个简单清晰而且统一的界面，从而维持了已有软件的兼容

性，极大地减轻了多核软件设计人员的负担。

下面我们将就集成电路设计中的可靠性挑战，即生产缺陷导致处理器核失效，以及工艺偏差导致处理器核性能不对称这两个问题，介绍多核处理器虚拟化技术的应用。

2 多核处理器虚拟化之拓扑重构技术

不断细化的半导体工艺对生产缺陷（Manufacturing Defects）越来越敏感，导致晶圆上能够正常工作的芯片所占的比例减少，即成品率的损失问题严重。而成品率是决定一款芯片产品成本和利润的主要因素。对于多核以及众核处理器来说，由于其面积较大，成品率损失问题十分严重。IBM 公司在 2006 年的一篇报告中指出[7]，像 Cell 这样的八核芯片，成品率只有 10%～20%。另一方面，使用先进工艺生产和制造的芯片，更易受到宇宙射线、电源噪声等干扰，导致软错误（Soft Error）及瞬时故障（Transient Fault），芯片上也更易发生电子迁移、热载流子退化、栅氧击穿等现象而导致永久故障（Permanent Fault），从而大大缩短了芯片的平均无故障时间，严重影响了芯片的寿命和可靠性。

为了提高微处理器的成品率，通常的做法是在微体系结构中增加冗余备份单元来替换失效的部分[8]，如修改流水线结构、为寄存器文件、缓存（Cache）之类增加冗余单元等。对于大规模多核处理器，相比为每一个核提供微体系结构级冗余的做法，核间冗余将成为更有效的缺陷容忍方法[9, 10, 11]。随着芯片上集成的处理器核的数目不断增多，单核将会变得非常精简，其所占的面积和整个芯片相比显得微不足道；同时，由于缺陷具有成簇分布的特点，使得芯片上失效的核只是很少的而且非常集中的一部分。因此，在工业界的多核处理器芯片中，也开始采用核间冗余技术提高芯片的成品率和性能，如 IBM 的 Cell 处理器、SUN 的 UltraSPARC T1 处理器[12]以及 Azul 的 Vega2 处理器[13]等。

在设计核间冗余的多核处理器以及众核处理器时有两种机制，分别为降级模式（degradation mode）（或者叫做 AMAA 模式（As Many As Available）），以及冗余模式（redundancy mode）（或者叫做 AMAD 模式（As Many As Demand））。降级模式也称为核备份，即当一个芯片上的某个处理器核失效后，便从系统中删除，只使用所有可用的处理器核。例如生产一款四核处理器芯片，当芯片上无失效的处理器核时，就是一个全功能的芯片；如果芯片上有一个、两个或者三个处理器核失效时，芯片将降级成为一个三核、双核或者单核芯片。如果所有的处理器核都失效，那么芯片将被丢弃，造成成品率的损失。冗余模式，又称之为“N+M”模式，是指为了生产一款 N 核处理器芯片，我们在芯片上另外多提供 M 个冗余的处理器核。我们总是为用户提供 N 个可用的处理器核。如果失效的核的数目小于 M，那么芯片可以被修复，同时芯片上存在一些没有使用的核；如果失效的核的数目大于 M，则按照降级模式使用。降级模式适用于目前的多核处理器，因为芯片上处理器核的数目还比较少，单核占据芯片的面积相对较大，为了使得开销最小，应该使用所有可用的处理器核。但是对于大规模多核处理器来说，由于芯片上集成了大量的处理器核，单核与整个芯片相比，面积和成本微乎其微，芯片上即使存在一些未使用的核，开销也是可以接受的。另外，由于芯片上的处理器核的数目众多，如果采用降级模式，那么我们会得到很多的降级版本，例如 100 核的芯片经过降级后，我们可能得到 99 核、98 核、97 核等等，而我们所需要的 100 核芯片的成品率却不能得到保证。最后从商业的角度看，如此多的降级版本会使得销售发生混乱，降低用户对芯片厂商的信心。因此对于大规模多核处理器来说，为了提高其成品率，应该采用冗余模式的核间冗余机制。冗余模式为用户透明地提供所需要的处理器核，保证了成品率，同时也不会导致市场的混乱。这也符合多核处理器虚拟化的基本思想。IBM 在它的报告中指出，由于 Cell 处理器（含有 8 个协处理器）的成品率较低，索尼的 PlayStation 3 游戏机将只使用其中的 7 个来提高产率，这其实就是“7+1”模式（N = 7, M = 1）。

芯片成品率的提高是以性能的降级为代价的。对于降级模式来说，导致性能损失的主要原因是处理器核的数目减少了。在冗余模式下的大规模多核处理器中，虽然计算能力能够得到保证，但是由于芯片在生产制造出来之前，哪些核失效哪些核能正常工作是未知的，因此核与核之间的互连拓扑关系是变化的，如图 3 所示。为了方便说明，假设我们要设计生产一款 9 核处理器芯片，拓扑结构是 3×3 的二维网状（2-DMesh）结构，如图 3(a)所示。为了提高其成品率，我们提供一列的冗余核作为备份，如图 3(b)所示。如果片上有失效核存在（不超过 3 个），那么我们仍然可以向用户提供 9 核芯片。然而如图 3(c) 所示，失效核不仅使得各个芯片的拓扑结构与目标拓扑结构（3×3 的二维网状）不同，而且失效核不同所形成的各个芯片的拓扑结构也各不相同。这些被改变的拓扑结构变得很不规整，从而导致了多核处理器芯片的性能降级。

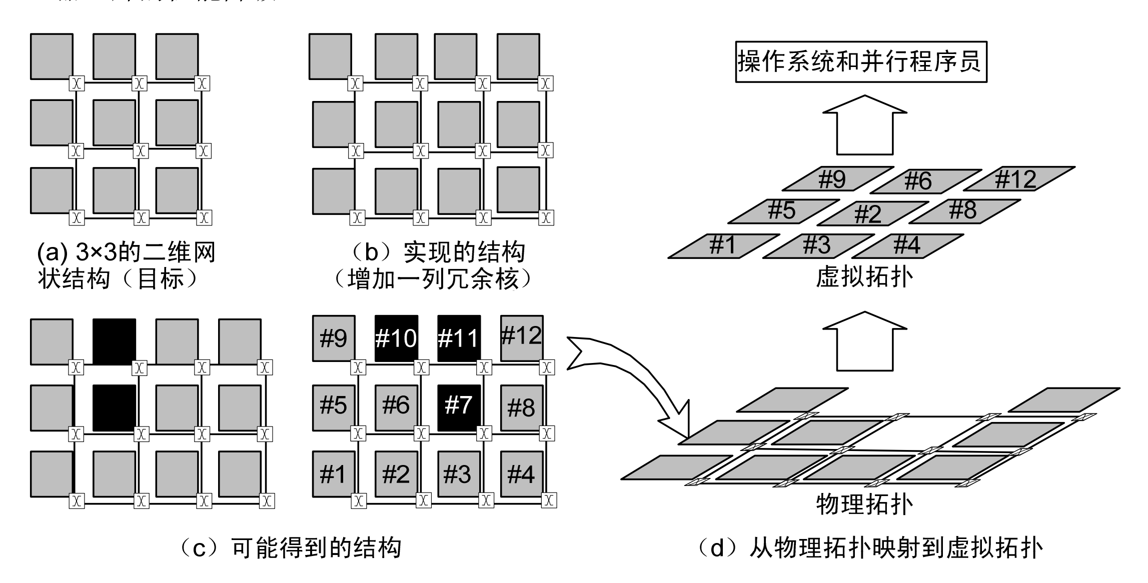


图3. 拓扑虚拟化

由于片上通信的性能极大地影响着并行程序，因此操作系统和并行程序员都需要了解底层的拓扑结构，以便进行任务的分配、调度以及程序的优化[16]。然而在冗余模式下，众核处理器的底层拓扑结构可能各不相同，操作系统和程序员需要针对各种不同的结构进行优化，而在一种拓扑结构上优化的程序在另一种结构上的性能可能很差。为了屏蔽复杂的硬件细节，我们可以将底层的"物理"拓扑结构虚拟化，如图 3(d)所示。我们可以屏蔽掉各种各样的底层物理拓扑结构。上层的操作系统和程序员看到的始终是一个统一的拓扑结构。同时，不管底层的处理器核是如何互连如何失效的，这个拓扑结构还是和参考拓扑结构同构的。这种映射和屏蔽极大地简化了操作系统对任务的调度和分配，同样程序员只需要针对参考拓扑结构进行应用软件的优化。由于众核处理器的拓扑结构需要提供给程序员，这种统一的拓扑结构界面还避免了销售市场的混乱。

从本质上讲 N+M 机制和上述的拓扑重构都属于多核虚拟化的思想。N+M 机制旨在为用户提供他们所需要的数目的处理器核而不管底层有多少可以工作的处理器核；拓扑重构则旨在为用户提供一个他们所需要的拓扑结构而不管底层的处理器核之间是如何互连的。N+M 机制和拓扑重构机制都是将各种各样的底层细节屏蔽掉，让用户始终面对一个清晰统一的界面，消除了混乱，简化了编程使用。

在 Cray T3E 机器中使用过类似的思想[14]。当系统运行时如果有处理器发生失效，那么可能有一些处理器在物理上就不连续了。为了向应用提供一个连续的逻辑处理器编号，系统

路由表和逻辑“who am I”寄存器允许节点进行逻辑重命名，也就是实现从物理节点到逻辑节点的重新映射。这种“热交换”对用户是完全透明的。然而这种热交换和我们的拓扑重构是不同的，因为对于像 Cray T3E 这种机器来说，节点失效和拓扑改变是暂时的，系统可以很容易地通过重新加入新的节点来修复。而对于众核处理器来说，缺陷造成的拓扑改变是永久的。

实现从物理拓扑结构到虚拟拓扑结构的映射有很多种方法。我们可以在芯片上增加一个映射表，记录虚拟处理器核与物理处理器核之间的对应关系。映射表以固件（firmware）的形式实现。在对芯片进行测试之后，就可以确定芯片的物理拓扑结构，也就是哪些核失效了，哪些核可用。根据一定的重构算法得到对应的虚拟拓扑结构，并将这些信息写入到映射表中，如图 4 所示。

当操作系统进行任务分配时，比如在虚拟处理器核#V 上分配了一个线程，固件会根据映射表信息将该线程分配到物理处理器核#2 上运行。当运行在虚拟处理器核#V 和#VI 上的线程之间进行通信时，固件根据映射表将通信双方的地址映射到物理处理器核#2 和#8。操作系统和程序员都不需要直接面对各种不同的物理结构进行优化和调度，他们只需要针对参考拓扑结构进行优化，其它的工作由固件来完成。这就极大地减轻了操作系统和程序员的负担。

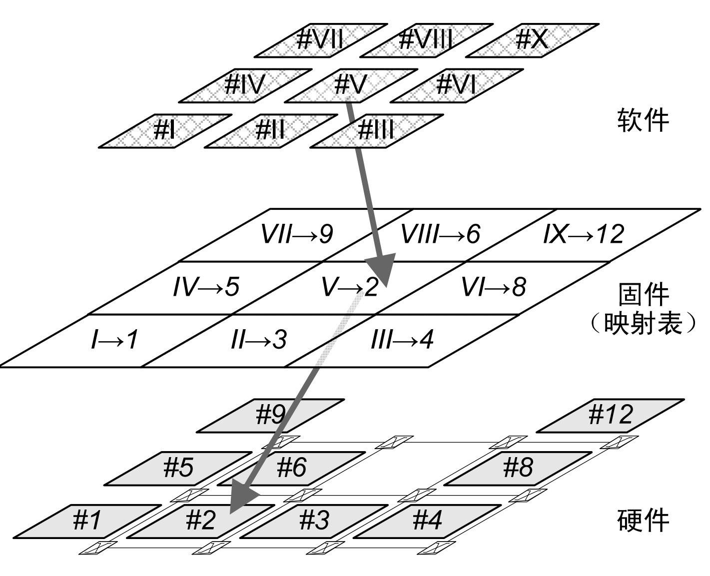


图4. 拓扑虚拟化的工作方式

一个物理拓扑可以映射出非常多的虚拟拓扑，因为我们并没有限制物理拓扑中无故障的处理器核应该放置在虚拟拓扑的哪个位置。例如对于图 4 中的物理拓扑结构可以映射得到 9！个可能的虚拟拓扑。由于虚拟拓扑结构会导致应用程序的性能降级，而一个给定的物理拓扑又能映射出非常多的虚拟拓扑，因此我们需要从众多的虚拟拓扑中选择一个对应用性能影响最小的提供给用户。虚拟拓扑和参考拓扑相比，性能降级的主要原因是虚拟拓扑在物理上变得不规整了，因而在重构拓扑的时候，应该尽可能地保证虚拟结构的规整性。如图 5 所示的行波列借算法正是基于这一原则，以行和列为单位维护拓扑的规整性。

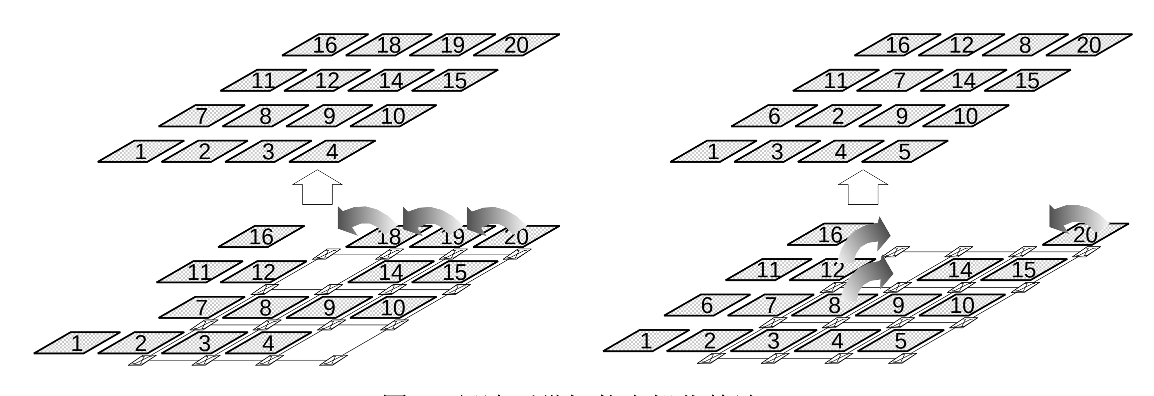


图5. 行波列借拓扑虚拟化算法

3 多核处理器虚拟化之核性能碎片整理技术

除了生产缺陷外，工艺偏差（process variation）也是当前芯片设计关注的一个重要问题。工艺偏差包括系统级工艺偏差和随机偏差两类。系统级偏差主要是由于光刻的析光差，并表现为一定的空间相关性，也就是两个晶体管之间是否存在工艺偏差，只取决于它们之间的距离。而随机偏差主要来源于掺杂的不均匀，因此对晶体管的影响是完全随机和独立的，也就是没有空间相关性[15]。

文献[17]指出当工艺特征尺寸为 130nm 时，系统级偏差和随机偏差对芯片的影响是相当的，而随着工艺的进一步细化，随机偏差将占主导地位，这也就意味着芯片上的晶体管属性的差异将只存在很少的空间相关性。工艺偏差对多核处理器的影响就是导致核与核之间的性能不对称，虽然在设计阶段处理器核具有相同的指令集和组织结构。文献[17]指出工艺偏差使得处理器核之间的频率差异能到达 20%。因此未来的多核处理器将会表现为下面两个特点：1） 不同处理器核的频率差异将变大；2）处理器核的频率分布也将更加随机。

在上一部分中介绍的拓扑虚拟化技术只将被改变的片上网络拓扑结构进行抽象，并且假设处理器核的性能都是相同的，因此对于性能不一致的多核处理器其使用将受到限制。我们下面举例说明。如图 6 所示，假设不同处理器核的性能存在差异。第一个虚拟拓扑能够保持较好的核间距离（1 跳），如果我们将中间的两个核进行交换， 那么一些核间的距离将增加。但是这样做的好处是所有高性能的处理器核都位于第一列。由于操作系统在进行任务分配和调度的时候都是基于“连续”的原则，即将任务分配在连续的处理器核上。当我们为操作系统提供虚拟拓扑 II 时，则有利于加速并行度较高的应用程序。例如，一个线程的程序在第二个虚拟拓扑结构上可以获得很好的加速，而对于第一个虚拟拓扑来说，任何并行度超过 1 的应用都会有性能的损失，因为低性能的处理器核将极大地降低高性能处理器核带来的好处。

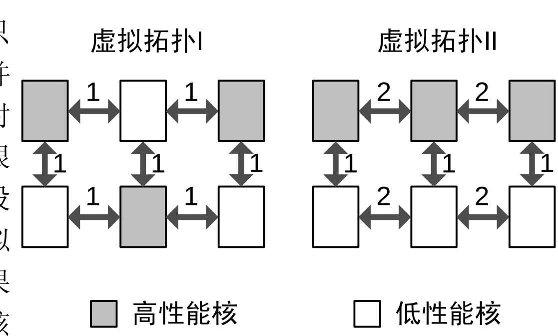
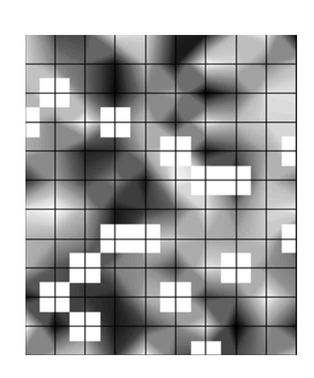
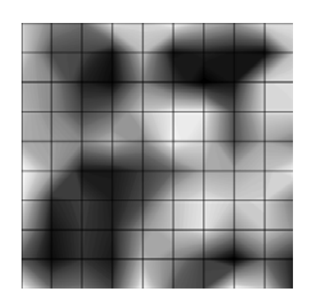


图6. 考虑核性能不对称的多核处理器虚拟化

这种现象和存储器的碎片整理技术要解决的问题非常类似。由于操作系统按页对存储器进行分配，在分配的过程中可能出现碎片（fragment）。这些碎片使得存储器中连续的存储区域减少，即操作系统可以分配的最大空间减少，从而增加了换入换出外围设备如硬盘的次数，降低了应用的性能。因此在存储器中，经常采用碎片整理技术，将页碎片进行整理和压缩，从而为操作系统提供一个连续的较大的可分配的空间。

同样在多核处理器中，由于工艺偏差导致处理器核的性能不对称，而低性能的处理器核会大大降低高性能处理器核带来的性能提高，因此在虚拟化中必须考虑核不对称的情况。我们可以将性能相近的处理器核成簇的分布，同时又注意不至将簇内核间的距离变得非常远。这样虚拟化后的多核处理器，操作系统无须改变已有的连续分配的调度方案，而拥有较多的连续的高性能处理器核，从而支持并行度较高的应用的加速。如图 7(a)所示为虚拟化之前的多核处理器，带有失效核（白色区域），核间存在性能差异（颜色深表示性能较高，颜色浅表示性能较差）。图 7(b)表示虚拟化之后的多核处理器，结构规则，同时性能相近的处理器核集中分布在一起，图中表示为连续的深色区域，当高并行度的任务分布在这些区域时，可以获得较高的性能提高。





（a）虚拟化前的多核处理器：带有失效核，核间性能不对称　（b）虚拟化后的多核处理器:拓扑虚拟化，核性能碎片整理

图7. 虚拟化前后的多核处理器比较

4 总结

多核处理器及众核处理器面临着严重的可靠性设计挑战，其中一个主要方面是这些可靠性问题，如生产缺陷，工艺偏差等会导致多核处理器底层的结构与最初的设计目标差异很大，而且不同芯片会表现为不同的特点和属性。这为多核处理器的系统软件及应用软件设计人员带来很大的负担，而从软件的层次去适应这些异构性将会非常复杂，并导致严重的性能损失。多核处理器的虚拟化技术可以有效地解决这一异构性问题。通过在硬件和固件层检测并配置，为上层软件提供一个清晰统一简单的界面，极大地减轻了系统软件的负担，同时使得软件具有较好的兼容性。本文针对两类主要的可靠性设计挑战，即生产缺陷导致处理器核失效从而改变核间互连拓扑的问题， 以及工艺偏差导致处理器核间性能不对称的问题，介绍了多核处理器的虚拟化方法。虚拟化方法还可以用于解决如双模/三模冗余以及功耗管理等其他要求所面临的多核处理器的异构性问题。

参考文献：

[1] D. Geer, “Chip makers turn to multicore processors,” *IEEE Computer*, vol. 38, no. 5, pp. 11–13, May 2005.

[2] S. Borkar, “Thousand core chips—A technology perspective,” in Proc. *ACM/IEEE Design Automation Conf. (DAC)*, Jun. 2007, pp. 746–749.

[3] Intel, “From a Few Cores to Many: A Tera-Scale Computing Research Overview” [Online]. Available: http://techresearch.intel.com/articles/Tera-Scale/1421.htm

[4] W. J. Dally and B. Towles. “Route Packets, Not Wires: On-chip Interconnection Networks”, in Proc. *ACM/IEEE Design Automation Conf. (DAC)*, pp. 18-22, June 2001.

[5] L. Benini and G. De Micheli. “Networks on Chips: a New SoC Paradigm”, *IEEE Computer*, vol. 35, no.1, pp. 70-78, Jan. 2002.

[6] P. M. Wells, K. Chakraborty, and G. S. Sohi. “Adapting to intermittent faults in multicore systems”. In Proc. of *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, 2008.

[7] Ed. Sperling, “Turn Down the Heat Please,” [Online]. Available: http://www.edn.com/article/CA6350202.html Mar. 2007

[8] I. Koren and Z. Koren, “Defect tolerance in VLSI circuits: Techniques and yield analysis,” *Proceedings of IEEE*, vol. 86, no. 9, pp. 1819–1838, Sep. 1998.

[9] P. Shivakumar, S.W. Keckler, C. R.Moore, and D. Burger, “Exploiting microarchitectural redundancy for defect tolerance,” in *Proc. IEEE Int. Computer Design Conf. (ICCD)*, Oct. 2003, pp. 481–488.

[10] E. Schuchman and T. N. Vijaykumar, “Rescue: A microarchitecture for testability and defect tolerance,” in *Proc. IEEE/ACM Int. Symp. Computer Architecture (ISCA)*, Jun. 2005, pp. 160–171.

[11] L. Zhang, Yinhe Han, Q. Xu, X. Li and H. Li, “On Topology Reconfiguration for Defect-Tolerant NoC-Based Homogeneous Manycore Systems”, *IEEE Transactions on VLSI Systems*, vol. 17, no.9,

2009
[12] P. J. Tan, T. Le, K.-H. Ng, P. Mantri, and J.Westfall, “Testing of Ultra-SPARC T1 microprocessor and its challenges,” in Proc. *IEEE Int. Test Conf. (ITC)*, Oct. 2006, Paper 16.1.

[13] S. Makar, T. Altinis, N. Patkar, and J. Wu, “Testing of Vega2, a chip multiprocessor with spare processors,” in Proc. *IEEE Int. Test Conf. (ITC)*, Oct. 2007, Paper 9.1.
[14] S. L. Scott and G. M. Thorson, “The Cray T3E Network: Adaptive routing in a high performance 3D torus,” in Proc. *Hot Interconnects*, Aug. 1996, pp. 147–156.
[15] S. R. Sarangi et al, “Varius: A model of process variation and resulting timing errors for microarchitects,” *IEEE Transactions on Semiconductor Manufacturing*, vol. 21, 2008.
[16] A. Grama, G. Karypis, V. Kumar, and A. Gupta, “Introduction to Parallel Computing (2nd Edition)”. Addison Wesley, 2003.
[17] T. Karnik, “Probabilistic and Variation-Tolerant Design: Key to Continued Moore’ s Law Scaling,” *Invited talk in ACM/IEEE International TAU Workshop on Timing Issues*, 2004.

作者简介

张磊： 中科院计算所，计算机系统结构重点实验室，助理研究员，zlei@ict.ac.cn
韩银和： 中科院计算所，计算机系统结构重点实验室，副研究员
李晓维： 中科院计算所，计算机系统结构重点实验室，研究员